

Cross-Framework Benchmarking of Quantum Support Vector Machines on MNIST

Ponze, Karina

Department of Computer Science/IT

Kean University

Union, NJ, USA

ponzek@kean.edu

Abstract—Quantum Support Vector Machines (QSVMs) have been explored as potential enhancements to classical classifiers, with performance often assessed within a single software framework. Here, we conduct a cross-framework benchmark of QSVMs using Qiskit, Cirq, and PennyLane on binary MNIST classification (digits 0 vs. 1). While the classical SVM baseline consistently achieves 99.0% accuracy across all frameworks, QSVM accuracy varies significantly: 45.0% (Qiskit), 71.5% (PennyLane), and 88.0% (Cirq). Moreover, QSVM training time spans three orders of magnitude (3.33 s–1057.90 s), compared to near-instant classical SVM training. These disparities arise not from quantum advantage, but from framework-specific integration overhead—highlighting that framework selection strongly affects QML benchmarking. We therefore emphasize the need for standardized benchmarks and transparent reporting to fairly assess quantum machine learning algorithms.

Index Terms—quantum machine learning, support vector machine, quantum kernel, Qiskit, Cirq, PennyLane, MNIST

I. INTRODUCTION

These quantum kernel methods go beyond classical support vector machines (SVMs) by exploring quantum feature spaces [1] and have seen a foundational implementation for quantum-improved feature maps in an exploration of MNIST [2] as well as a range of QSVM works comparing different model types. However, empirical cross-framework studies are lacking. The discrepancy is hidden by implementation details that drive performance differences between the frameworks, not quantum issues.

Previous work exploring the capacity of customary QSVMs has either focused on a single framework Slyszy et al. [2] tested Qiskit on the MNIST handwritten character dataset after resizing) or on a comparison between variations of quantum models. For example, Villalba-Ferreiro et al. [1] compared QSVC and QNN implementations from Qiskit and PennyLane, respectively, or avoided framework comparison altogether by executing hybrid autoencoder-QSVM pipelines (Slabbert and Petruccione [3]). However, the descriptions of the tools were not benchmarked by Shrinivas et al. [4].

To that end, we benchmark a QSVM and classical SVM implementation in three of the most popular quantum machine learning frameworks: Qiskit, Cirq, and PennyLane.

- 1) Experimental evidence shows the classical SVM baseline accuracy is affected more by the framework than by the quantum model.

- 2) Trade-off between training time and accuracy for various frameworks on subsets of MNIST.
- 3) Show that the quantum simulation speed advantage (<0.02 s) cannot be extended to actual hardware without reducing noise.

II. METHODOLOGY

A. Dataset and Preprocessing

We used the Noisy Intermediate-Scale Quantum (NISQ)-era preprocessing of the MNIST dataset [3]. For our binary classification task (0 vs. 1), we have a total of 14,780 samples. The features were scaled with StandardScaler, then the data were embedded in 2D with PCA for 2-qubit amplitude encoding, ensuring class separability [2]. The dataset was then separated into 100/200 training/test sets after scaling with the quadratic kernel [1].

B. Classical SVM

These baselines were trained on an RBF kernel configuration, with $C=1.0$ in scikit-learn. Frameworks were selected based on Shrinivas et al. [4], which compared similar state-of-the-art tools.

C. Framework Implementations

Three frameworks were explored and run using the same hyperparameters:

- **Qiskit**: Aer simulator with ZZFeatureMap (2 repetitions), FidelityQuantumKernel computed via ComputeUncompute (full circuit simulation).
- **Cirq**: TensorFlow Quantum backend, RY+entangling feature map, CZ gates for inter-node connectivity
- **PennyLane (NumPy simulator)**: ZZFeatureMap (repeated twice), fidelity kernel via state overlap without circuit simulation overhead

D. Metrics

Accuracy (test-set accuracy) and wall-clock time were recorded on a Google Colab runtime, where the same hardware was used for all measurements. Kernels in this way are the quantum fidelity between pairs [2]:

$$K(\mathbf{x}_i, \mathbf{x}_j) = |\langle 0 | U^\dagger(\mathbf{x}_i) U(\mathbf{x}_j) | 0 \rangle|^2 \quad (1)$$

where $U(x)$ is the parameterized quantum feature map, which encodes classical input features x into a quantum state by applying a sequence of rotation and entangling gates. In our implementation, $U(x)$ is applied twice, and corresponds to the ZZFeatureMap. R_Z rotations parameterized by the input features and controlled- Z entangling gates generate quantum-improved feature representations in the Hilbert space. The adjoint operation $U^\dagger(x_i)$ the "compute-uncompute" protocol, to estimate the state overlap fidelity. $|\langle 0|U^\dagger(x_i)U(x_j)|0\rangle|^2$, the value of the quantum kernel between the two samples x_i and x_j . This fidelity-based kernel replaces the classical radial basis function (RBF) kernel in support vector classification, and it theoretically allows classically nonlinearly separable data to be separated in quantum feature space embeddings [2].

III. RESULTS

The accuracy and training time results are shown in Table I. Key observations include:

- The QSVM accuracy varied substantially by framework: 71.5% in PennyLane, 88.0% in Cirq, and only 45.0% in Qiskit. The low performance in Qiskit is attributed to the computational overhead of the ComputeUncompute method during kernel matrix evaluation—an implementation-specific bottleneck rather than a limitation of the quantum model itself.
- In contrast, the classical SVM baseline achieved 99.0% accuracy consistently across all frameworks, indicating that algorithmic performance was stable and that observed QSVM variations stem from framework integration artifacts.
- Classical SVM training completed in under 0.02 s in all cases, while QSVM training time ranged from 3.33 s in PennyLane's lightweight simulation environment to 1057.90 s under Qiskit's full circuit simulation—a difference of three orders of magnitude.

TABLE I
QSVM VS CLASSICAL SVM PERFORMANCE ACROSS FRAMEWORKS

Framework	Model	Accuracy (%)	Time (s)
PennyLane (NumPy)	Classical SVM	99.0	0.00
	QSVM	71.5	3.33
Cirq	Classical SVM	99.0	0.02
	QSVM	88.0	50.27
Qiskit	Classical SVM	99.0	0.00
	QSVM	45.0	1057.90

Although quantum kernel evaluation for a single data pair can be fast (<0.02 s in simulation), full QSVM training time ranged from 3.33 s to 1057.90 s across frameworks, which is far slower than classical SVM training who completed in under 0.02 s in all cases. This large discrepancy is especially pronounced in Qiskit, where the ComputeUncompute kernel evaluation method incurs heavy simulation overhead.

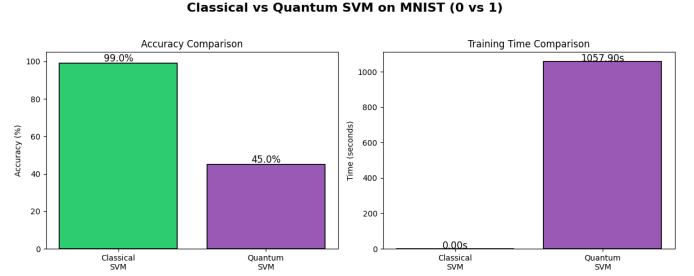


Fig. 1. Accuracy and training time comparison for Classical SVM and Qiskit QSVM.

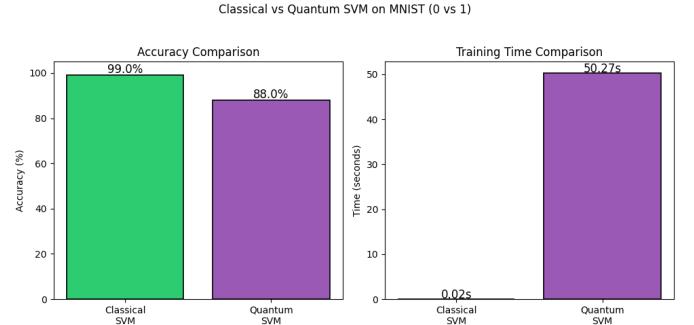


Fig. 2. Accuracy and training time comparison for Classical SVM and Cirq QSVM.

IV. DISCUSSION

Classical SVM training is consistently fast (<0.02 s) and accurate (99%), while QSVM performance varies widely across frameworks; both in accuracy (45.0% – 88.0%) and in training time (3.33 s – 1057.90 s). These differences are framework-specific and not intrinsic to the quantum model. The 43-percentage-point accuracy gap between the lowest and highest QSVM results (45% vs. 88.0%) suggests that implementation details, such as kernel evaluation overhead, dominate variations in quantum kernel quality.

Thus, we support Villalba-Ferreiro et al.'s conclusion that "framework choice impacts QSVM benchmarking" [1], and further note that such framework effects also influence classical baselines, though to a lesser extent. To ensure reproducible QML research, we recommend:

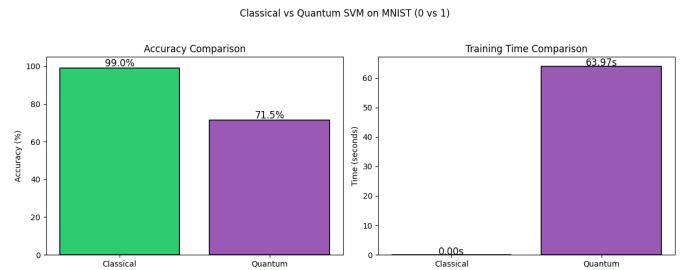


Fig. 3. Accuracy and training time comparison for Classical SVM and PennyLane QSVM.

- 1) Reporting framework version, simulator or hardware backend, and kernel computation method.
- 2) Using identically configured classical baselines across frameworks.
- 3) Explicitly accounting for integration overhead as a confounding factor.

Current NISQ limitations, including finite sampling and simulation-only execution, affect all experiments. For example, Qiskit's QSVM achieves only 45% accuracy because its `ComputeUncompute` method simulates full circuits for each kernel entry, a known scalability issue [2]. Future work should test these algorithms on real hardware under error mitigation to better understand performance in realistic settings.

V. CONCLUSION

Cross-framework benchmarks show that the implementation of the quantum software stack heavily impacts the performance of QML algorithms. The results indicate that QM models (QSVM) have a highly variable range of accuracy (45.0% to 88.0%) and training time (3.33 s to 1057.90 s). In contrast, classical SVM baselines achieve consistent accuracy (99%) with a low training time (<0.02 s).

This underlines that QML benchmark performance is typically more influenced by intrinsic factors of the framework, like using kernel computation methods or the precision of simulation, than originating from quantum advantage. We recommend that quantum machine learning research should adopt consistent benchmarking, open reporting of framework options for reproduction, and separating the algorithmic promise from implementation artifacts.

VI. ACKNOWLEDGEMENT

Code and experimental results are openly available at github.com/ponzek/MNIST_ClassicalMLvsQML.git.

REFERENCES

- [1] T. Villalba-Ferreiro, E. Mosqueira-Rey, and D. Alvarez-Estevez, "Performance analysis of quantum support vector classifiers and quantum neural networks," arXiv:2512.03094, 2025.
- [2] M. Słysz, K. Kurowski, G. Waligóra, and J. Węglarz, "Exploring the capabilities of quantum support vector machines for image classification on the MNIST benchmark," in *Int. Conf. Comput. Sci.*, 2023, pp. 185–197.
- [3] D. Slabbert and F. Petruccione, "Classical-quantum approach to image classification: Autoencoders and quantum SVMs," *AVS Quantum Sci.*, vol. 7, p. 023804, 2025.
- [4] S. Shrinivas, H. S, and M. A, "Comparison of quantum machine learning tools," Thiagarajar College Eng., 2024.